> EXPRESS MAIL LABEL NO.:
>
> (EL708269755US)

VIRTUAL PATH RESTORATION USING BEST EFFORT TECHNIQUES

Haig Michael Zadikian

Zareh Baghdasarian

Ali Najib Saleh

Vahid Parsi

CROSS-REFERENCES TO RELATED APPLICATIONS

This application relates to the following commonly assigned applications:

1.   Patent Application Serial No. 09/751,763; filed December 30, 2001, entitled, "A METHOD FOR ROUTING INFORMATION OVER A NETWORK EMPLOYING CENTRALIZED CONTROL," (Attorney Docket No. M-8575 US).

2.   Patent Application Serial No. 09/859,166; filed May 16, 2001, entitled, "A METHOD FOR RESTORING A VIRTUAL PATH IN AN OPTICAL NETWORK USING 1:N PROTECTION," (Attorney Docket No. M-7165-8P US).

The above mentioned applications are assigned to Cisco Technology, Inc., the assignee of the present invention, and are hereby incorporated by reference in their entirety and for all purposes.

BACKGROUND OF THE INVENTION

Field of the Invention

This invention relates to the field of communication networks, in particular to a method and apparatus to re-establish communication links after communication links
5   experience a failure.

DESCRIPTION OF THE RELATED ART

Today's networks carry vast amounts of information. High bandwidth applications
10   supported by these networks include streaming video, streaming audio, and large aggregations of voice traffic. In the future, these bandwidth demands are certain to increase.

To meet such demands, an increasingly popular alternative is the use of light wave communications carried over fiber-optic cables. The use of light wave communications provides several benefits, including high bandwidth, ease of installation, and capacity for future growth.

5   Optical infrastructures are capable of transmission speeds in the gigabit range, which helps address the ever-increasing need for bandwidth mentioned above. Such infrastructures employ various topologies, including ring and mesh topologies. In order to provide fault protection, ring topologies normally reserve a large portion (e.g., 50% or more) of the network's available bandwidth for use in restoring failed circuits. However, ring topologies

10   are capable of quickly restoring failed circuits. This capability is important in providing reliable service to customers, and is particularly important in telephony applications, where a failure can result in alarms, dropped calls, and, ultimately, customer dissatisfaction and lost revenue. In a similar vein, because of bandwidth demands, protocol overhead related to provisioning, restoration, and other functions should be kept to a minimum in order to make

15   the maximum amount of bandwidth available for use by customers.

An alternative to the ring topology, the mesh topology reduces the amount of bandwidth needed for protection. The mesh topology is a point-to-point topology, with each node in the network connected to one or more other nodes. Because a circuit may be routed through various combinations of the network's nodes and over the various links which

20   connect them, excess capacity through a given node or over a given link can serve to protect several circuits. The restoration of a circuit following a failure in a mesh topology can consume a relatively large amount of time.

Therefore, there is the tradeoff in ring topologies that can restore communication quickly but take up a great deal of bandwidth, and mesh topologies that do not take up as

25   much bandwidth but are much slower in restoring communications. Current communication networks provide continuous, and as users have become accustomed to, uninterrupted transmission. A need therefore has been felt for a method and apparatus that allows for rapid restoration of communication in the event of the failure of a link.

30   **SUMMARY OF THE INVENTION**

What is needed and is disclosed herein is an invention that provides for restoration of paths between zones, where the zones include a number of nodes. A zone can have a source node or the zone can have a destination node. Border nodes connect to other border nodes of

other zones. The zones connected with one another make up a higher hierarchical zone referred to herein as a backbone zone.

In some embodiments, in order to reduce the time to restoration of failed links between a source and a destination node, a determination is made as to alternative routes that may exist between nodes. Some embodiments can employ metrics such as class of service, which are used to determine the allocation of resources (i.e. links, bandwidth, or the like) between source and destination nodes. Further, the backbone zone can be preplanned which provides further reduction of the time of restoration of failed links.

The foregoing is a <u>summary</u> and thus contains, by necessity, simplifications, generalizations and omissions of detail; consequently, those skilled in the art will appreciate that the summary is <u>illustrative only</u> and is <u>not</u> intended to be in any way <u>limiting</u>. Other aspects, inventive features, and advantages of the present invention, as defined solely by the claims, will become apparent in the non-limiting detailed description set forth below.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention may be better understood, and it's numerous objects, features and advantages made apparent to those skilled in the art by referencing the accompanying drawings. The use of the same reference number throughout the figures designates a like or similar element.

Figure 1 illustrates a backbone zone.

Figure 2 illustrates a nodal zone of a backbone zone.

Figure 3 illustrates inter-zone communication.

Figure 4 is a flowchart illustrating setup of the shortest path formula algorithm table.

Figure 5 is a flowchart illustrating how the shortest path formula algorithm looks up the shortest path.

Figure 6 is a block diagram illustrating a network environment in which embodiments of the present invention may be practiced.

Figure 7 is a block diagram illustrating a computer system suitable for implementing embodiments of the present invention.

Figure 8 is a block diagram illustrating the interconnection of the computer system of Figure 7 to client and host systems.

While the invention is susceptible to various modifications and alternative forms, specific embodiments thereof are shown by way of example in the drawings and will herein be described in detail, it should be understood, however, that the drawings and detailed description thereto are not intended to limit the invention to the particular form disclosed but

5 on the contrary, the intention is to cover all modifications, equivalents, and alternatives falling within the scope of the present invention as defined by the appended claims.

DETAILED DESCRIPTION

A topology database can be used to provide information to nodes in a network

10 regarding connectivity of those nodes to other of those nodes and zones. Broadcast packets are sent by nodes whenever a failure occurs, effectively requesting the availability of other nodes to connect and establish a communication path. To limit the size of the topology database and the scope of broadcast packets, networks employing the protocol described herein can be divided into smaller logical groups called "zones." Each zone executes a

15 separate copy of the topology distribution algorithm, and typically nodes within each zone are only required to maintain information about their own zone. There is no need for a zone's topology to be known outside that zone's boundaries, and nodes within a zone need not be aware of the network's topology external to their respective zones. A network includes a number of nodes.

20 Nodes that attach to multiple zones are referred to herein as border nodes. Each zone has at least one border node, and that border node is coupled to at least one other border node of another zone. Border nodes are typically required to maintain a separate topological database, also called link-state or connectivity database, for each of the zones to which they are attached. Border nodes use the connectivity database for intra-zone routing. Border

25 nodes are also required to maintain a separate database that describes the connectivity of the zones themselves. This database, which is referred to herein as the network database, is used for inter-zone routing. The network database describes the topology of a special zone, referred to herein as the backbone zone. In certain embodiments, the backbone zone is always assigned a hierarchical identification (ID) of 0. The backbone has the characteristics

30 of a zone. There is no need for a backbone's topology to be known outside the backbone, and a zone's border nodes need not be aware of the topologies of other zones.

Fig. 1 is a block diagram illustrating a topology of a backbone zone. A zone 100 (also referred to as Zone 1) directly connects to a zone 110 (Zone 2). A zone 120 (Zone 3) is connected to zone 100 (Zone 1) and zone 110 (Zone 2). Zone 120 (Zone 3) indirectly

connects zone 100 (Zone 1) and zone 2 (Zone 110). In this particular example, the backbone zone 130 is referred to as Zone 0.

Fig. 2 is a block diagram illustrating a topology of a zone of a backbone zone. Zone 1 includes a number of nodes and links. In this particular example, "Zone 1" includes a node 240 (Node 1); a node 250 (Node 2); a node 260 (Node 3); a node 210 (Node 4); a node 220 (Node 5); a node 200 (Node 6); and a node 230 (Node 7). Links interconnect the nodes, and in this particular example, the topology includes a link 205, a link 215, a link 225, a link 235, a link 245, a link 255, a link 265, a link 275, and a link 285. In an embodiment, a link can be define as a logical group of one or more ports that connect two adjacent nodes (e.g., a physical interface). A port is a physical interface. There can be more than one link between adjacent nodes.

Within a zone, nodes can be distinguished by the attributes they possess. In one embodiment, the location of the node can determine the attributes of the node. A master node is defined as the endpoint of a link with numerically lower node ID. A master border node is defined as the end-node of an inter-zone link that is also a source node or proxy source node of one or more virtual paths (VP) that use that inter-zone link. A VP is an end-to-end connection with which is associated certain information such as a path bandwidth, class of service (CoS), quality of service (QoS) level, and the like. An inter-zone VP is one that traverses two or more zones. As described in the previously referenced patent application entitled "A METHOD FOR ROUTING INFORMATION OVER A NETWORK EMPLOYING CENTRALIZED CONTROL," the wavelength routing protocol (WaRP™) protocol describes a master border node as generating a *Create Path* request when the inter-zone link fails. A slave border node is the end-node of an inter-zone link that is also the destination or proxy destination node of one or more VPs that use that link. An entry border node is a border node that receives the *Create Path* request from an adjacent zone. An exit border node is a border node that forwards the *Create Path* request to an adjacent zone. An origin node is the origin of a WaRP™ packet (e.g. *Restore Path, Delete Path,* and *Test Path* packets). An origin node is either the source node of the VP or a proxy border node. In the case of an intra-zone failure, an origin border node is a border node that assumes the role of a source node during a path restoration attempt, and is responsible for generating the *Restore Path* request on behalf of the source node. The ID of the origin border node is carried in the origin field of the *Request Path* request.

Fig.3 illustrates a topology of inter-zone communication. Zone 100 (Zone 1) 100 is connected to zone 110 (Zone 2) by a link 300 (Link 0). The border node 240 (Node 1) of zone 100 (Zone 1) is connected by link 300 (Link 0) to border node 310 (Node 2) of zone 110 (Zone 2). The following naming convention will henceforth be used to describe a node. The

5   naming convention consists of the zone, followed by a period, and the node that is referred to within the specific zone. Therefore, node 240 (Node 1) of zone 100 (Zone 1) can also be referred to as Node 1.1. Node 310 (Node 2) of zone 110 (Zone 2) can also be referred to as Node 2.2.

In this particular example, inter-zone link "Link 0" 300 fails. When an inter-zone link

10   fails, or one of its two end nodes fail, the WaRP™ protocol uses a combination of broadcast and source-routed packets to reroute traffic around the failure.

In certain implementations, the WaRP™ protocol allows a single inter-zone failure to be restored within 50 milliseconds (ms) or less. In one embodiment, timely restoration (within 50 ms) during a second inter-zone failure can be affected by the WaRP™ protocol

15   algorithm using information contained in the topology database of the backbone zone, or in this example Zone 0, to compute new inter-zone routes for the failed VPs. Source routed packets are used to request and establish the new routes. In other words, no flooding or broadcasting of packets is ever attempted nor allowed between zones, only within zones or intra-zone. One of the two nodes on either end of the failed link that node being a master

20   node computes a shortest path first alternative for each failed route, and places the newly calculated routes into a *Create Path* packet, and sends the *Create Path* packet to the next backbone node along the path. Tandem border nodes then use the computed route to forward the packet toward its ultimate destination. Routes within each zone are established using the same flooding mechanism as described earlier. The basic flooding mechanism involves each

25   packet being sent to all active neighbors except the one from which the packet was received.

Intra-zone restoration activities preferably occur in parallel and proceed independently of one another. While routes are established, a second failure along an inter-zone link results in a negative response generated by one of the tandem border nodes. That negative response is propagated all the way to the master border node, and causes the master

30   border node to compute a new route for the VP and retry the operation or link. In most cases, this process increases the restoration time of the VP to over 100 ms (or the time required for 2 attempts). This lengthy restoration time can be avoided, and restoration times limited to 50 ms or less by pre-planning the backbone route for all inter-zone link failures. Only the

backbone route, the backbone route being made up entirely of inter-zone links, needs to be pre-planned. The one or more intra-zone sub-paths of the end-to-end route are still established dynamically using the *Restore Path* packet/request.

Restoration times can also be limited by eliminating any possibility of back-to-back inter-zone link failures. One way to deal with inter-zone link failures is to use traditional protection schemes like diverse routing (the use of physically dissimilar cabling and hardware) and self-healing rings (SHR). This is also known as providing redundant paths. Protecting inter-zone links can be justified because inter-zone links make up a very small percentage of the overall fiber capacity. Moreover, in some situations, there is not enough connectivity among zones to make mesh restoration in the backbone zone any more efficient than diverse routing and SHR.

One of the attributes that makes mesh restoration superior to other traditional schemes is mesh restoration's ability to allow for sharing capacity. The amount of capacity sharing, however, is highly dependent on the topology of the network, the richness of its connectivity, and the end-to-end demand requirements. For a sparsely connected network, such as may be the case in a backbone zone, capacity sharing is minimal. For such topologies, where connectivity is limited and a hop-count is small), the additional cost of using traditional restoration methods can be justified (a hop is the path between two network nodes, and the hop-count is the number of hops between a given pair of nodes. For example, a "two hop" route involves three nodes and two links a two links.

SHR provides very fast restoration of failed links by using redundant links between the nodes of each ring. Each ring consists of two rings, a ring supporting information transfer in a "clockwise" direction and a ring supporting information transfer in a "counter-clockwise" direction. The terms "east" and "west" are also commonly used in this regard. Each direction employs its own set of fiber optic cables, with traffic between nodes assigned a certain direction (either clockwise or counter clockwise). If a cable in one of these sub-rings is damaged, the ring "heals" itself by changing the direction of information flow from the direction taken by the information transferred over the failed link to the sub-ring having information flow in the opposite direction.

The detection of such faults and the restoration of information flow thus occur very quickly, on the order of 10 ms for detection, and 50 ms for restoration for most ring implementations. The short restoration time is critical in supporting applications, such as telephone networks, that are sensitive to QoS. Other applications that may be QoS sensitive include systems that require short restoration times to prevent old digital terminals and

switches from generating and initiating alarms, such as carrier group alarms. Alarms are undesirable because such alarms usually result in dropped calls, causing users down time and aggravation.

The protection bandwidth can be a user-configurable parameter, attaching a QoS metric to configured connections and links. The QoS parameter allows the amount of required spare capacity to be reduced even further, while maintaining the same quality of service for those connections that need it and, more importantly, can afford such treatment. In other words, high availability is mapped into a cost metric and only made available to users who can justify the cost.

It will be noted that, typically, restoration times that exceed 10 seconds can lead to timeouts at higher protocol layers, while those that exceed one minute can lead to disastrous results for the entire network. However, the price of such quickly-restored information flow is the high bandwidth requirements of such systems. By maintaining completely redundant sub-rings, an SHR topology requires 100% excess bandwidth. As noted, an alternative to the SHR topology is the mesh topology.

Networks based on mesh-type restoration are inherently more capacity-efficient than ring-based designs, mainly because each network link can potentially provide protection for fiber cuts on several different links. By sharing the capacity between links, a network using a mesh topology can provide redundancy for failure restoration at less than 100% of the bandwidth capacity originally required. Such networks are even more efficient when traffic transits several links. Using the described approaches, however, result in restoration times ranging from several minutes to several months.

Once the user has defined the topology of the network, the user can configure one or more connections between nodes. Each configured connection defines a virtual path between the two end points, which are not required to be direct neighbors or even belong to the same zone. Similar to a physical point-to-point connection, the resulting VP has an associated capacity and an operational state.

The two end points of a VP can be designated as having a master/slave relationship. The master node is also referred to herein as the *source node* of the VP, and the slave node is referred to herein as the *destination node*. The source node typically assumes recovery responsibilities for the VP and originates *Restore Path* requests. The destination node waits for a message from the source node informing the destination node of the new path to use for the connection.

The method in which VPs are restored is the same regardless of how backbone routes are obtained. If 1:1 protection is used in the backbone zone, the alternate route is simply the protection channel assigned to the failed span. For description of 1:1 and 1:N protection, see reference patent application entitled "A METHOD FOR RESTORING A VIRTUAL PATH IN AN OPTICAL NETWORK USING 1:N PROTECTION."

When mesh restoration is used, however, the route is computed automatically by running an shortest path first (SPF) algorithm on the backbone zone to find the shortest path between the two border nodes. The alternate route, regardless of how it is computed, is then placed in the *Create Path* request and sent to the target node.

Shortest Path First (SPF) algorithm

Routes can be computed using a QoS-based shortest-path algorithm or the SPF algorithm. The route selection process relies on configured metrics and an up-to-date view of the topology to find the shortest paths between any two nodes. The topology database contains information about all network nodes, their links, and available capacity.

All nodes are assigned globally unique IDs. This gives the user control over the master/slave relationship between nodes. The network detects duplicate IDs when node adjacency is established. All nodes found with a duplicate ID are disabled by the protocol. An appropriate alarm can be generated to provide notification of the problem so that proper action can be taken.

The SPF algorithm uses the following variables:

1- *Ready* - A queue that holds a list of nodes, or vertices, that need to be processed.

2- *Database* - The pruned copy of the topology database, which is acquired automatically by the node using the Hello protocol, the Hello protocol is described below. The computing node removes all vertices and or links that do not meet the specified QoS and bandwidth requirements of the route.

3- *Neighbors [A]* – An array of 'A' neighbors. Each entry contains a pointer to one of the neighbor data structures described below:

In certain embodiments, network nodes use a protocol such as that referred to herein as the Hello Protocol in order to establish and maintain neighbor relationships, and to learn and distribute link-state information throughout the network. The protocol relies on the periodic exchange of bi-directional packets, known as Hello packets, between neighbors. During the adjacency establishment phase of the protocol, which involves the exchange of INIT packets, nodes learn information about their neighbors, such as that listed in Table 1.

Parameter	Usage
Node ID	Node ID of the sending node, which is preferably, from 8 bits to 32 bits.
HelloInterval	How often Hello packets should be sent by the receiving node
HelloDeadInterval	The time interval, in seconds, after which the sending node will consider its neighbor dead if a valid Hello packets is not received.
LinkCost	Cost of the link between the two neighbors. This may represent distance, delay or any other metric.
LinkCapacity	Total link capacity
QoS3Capacity	Link capacity reserved for QoS 3 connections
QoSnCapacity	Link capacity reserved for QoS 0-2 connections

Table 1

Every WaRP™ protocol node is required to establish an adjacency with its immediate

5   neighbors. The adjacencies are used to exchange WaRP™ protocol packets with those

neighbors and to determine their operational status. Each adjacency is represented by a

neighbor data structure that contains all information needed to establish and maintain the

adjacency. Table 2 illustrates a minimum number of fields to establish neighbor data

relationship.

10

State	The state of the adjacency
NodeID	Node ID of the neighbor
InactivityTimer	A one-shot timer whose expiration indicates that no Hello packet has been seen from this neighbor since the last HelloDeadInterval seconds.
HelloInterval	This is how often the neighbor wants us to send Hello packets.
HelloDeadInterval	This is how long the neighbor wants us to wait before declaring it is dead when it stops sending Hello packets.
LinkControlBlocks	A list of all links that exist between the two

	neighbors.
Advertised LSAs	A list of LSAs received from this neighbor
LSAsToBeSent	A list of LSAs that need to be sent to this neighbor

Table 2

The link state advertisement (LSA) contains a list of the node's neighbors, links, the
5   capacity of those links, the QoS available over the links, one or more costs associated with
each of the links, and other pertinent information.

4- *Path* [N][H] - A two-dimensional array (N rows by H columns, where N is the
number of nodes in the network and H is the maximum hop count). Position (n, h)
10   of the array contains a pointer to the following structure illustrated in Table 3.

Cost	Cost of the path from R to n
NextHop	Next node along the path from R to n
PrevHop	Previous node along the path from n to R

Table 3

15   Now referring to Figure 4 illustrated is a flowchart that creates the SPF two-
dimensional array. The flowchart begins with **R**, the root node 400. The root node is the
node that computes the routes. Initially, column 1 of the array is filled 402. For each node **n**
that is known to **R**, entry *Path*[**n**][**1**] is initialized 404. A decision is made to determine if **n**
is a neighbor of **R** 406. If **n** is not a neighbor of **R** then the entry *Path*[**n**][**1**] is set to the
20   following values 408:

$$Cost = \text{MAX\_COST}$$
$$NextHop = \text{INVALID\_NODE\_ID}$$
$$PrevHop = \text{INVALID\_NODE\_ID}$$

The process follows by setting the variable *LastHop* to equal the column value **h** 424 and
25   then ending the process 426.

If **n** is a neighbor of **R** filling the two-dimensional array proceeds as follows. The entry *Path*[**n**][**1**] is set to the following values 412:

$$Cost = Neighbors \ [\mathbf{n}].LinkCost$$

$$NextHop = \mathbf{n}$$

$$PrevHop = \mathbf{R}$$

The node **n** is then placed in the *Ready* queue 414. The process then continues to fill the remaining columns [2] to [H] 416. A determination is made to see if the *Ready* queue is empty 418. If the *Ready* queue is empty, the process follows by setting the variable *LastHop* to equal the column value h 424 and then ending the process 426.

If the *Ready* queue is not empty the process continues. The column [h-1] is copied to the current column [h] 428. The process continues for each node that is in the *Ready* queue 430, and for each neighbor node **m** of **n** according to **n**'s LSA. The variable *ComputedCost* is calculated as the cost of the Path **R** to **n** plus the cost of the link between **n** and **m**. A determination is then made as to whether *ComputedCost* is less than the cost of the Path of **R** to **m**. If the *ComputedCost* is not less than the cost of the Path of **R** to **m** then the process follows by setting the variable *LastHop* to equal the column value **h** 424 and then ending the process 426.

If *ComputedCost* is less than the cost of the Path of **R** to **m**, the following values are set for entry *Path*[**m**][**n**] 440.

$$Cost = ComputedCost$$

$$NextHop = Path[\text{n}][\text{n-1}].NextHop$$

$$PrevHop = \mathbf{n}$$

The node **m** is then added to the *Ready* queue 442. The variable *LastHop* is set to equal the column value **h** 424 and the process is ended 426, if all nodes in the *Ready* queue are completed 430, and for all neighbor **m** nodes of **n** 432.

When the two-dimensional array is filled, for any given hop-count (1 through *LastHop*), the entry *Path*[][] contains the best route from **R** to all other nodes in the network. To find the shortest hop path from **R** to **n**, row **n** is searched until an entry with a cost not equal to the value of variable MAX_COST is found. To find the least-cost path between **R** and **n**, regardless of the hop count, entries 1 through *LastHop* of row **n** are scanned, and the entry with the lowest cost is selected.

Now referring to Figure 5, shown is a flow chart that illustrates how to determine the shortest route to a given node. The target node is set to node **n**, and the root node is set to **R**; *route[]* is the route between **R** (root) and **n** 502. A column is selected from row **n** of

Path[][], for example column **h** is chosen. A determination is made as to whether the column is greater than "1" 506. If the column is not greater than "1" the process is ended 510. If the column is greater than "1" the following values are set 508:

$$\mathbf{n} = Path[\text{n}][\text{h}].PrevHop$$

$$\text{Route}[\text{h-2}] = \mathbf{n}$$

$$\text{h} = \text{h} - 1.$$

Process 508 continues until the column value is not greater than 1.

An alternative embodiment to the above mentioned algorithm replaces the 2-dimensional array with a single column that contains the best path, regardless of hop count, to any given node. The basic algorithm remains the same except setting Route[h-2] to **n** is eliminated. All references to *Path[node][hop]* are replaced with *Path[node]*. The modified algorithm executes about eight times faster but provides a single route to each destination. The multi-hop result of the original algorithm can be emulated by calling the modified algorithm multiple times, once for each desired route. This requires that certain arcs and vertices be removed from the topology table prior to each call. Otherwise, the SPF algorithm would return the same path every time. Deciding which arcs and/or vertices to remove from the database depends on the objective being achieved. If the intent is to obtain two or more link-disjoint routes, then all arcs present in the first N routes should be disqualified prior to making the $(N+1)^{st}$ call. If the intent, however, is to obtain K distinct routes, then disqualifying one link from each computed route is sufficient to achieve the desired result.

The *Create Path* packet is used to restore VPs disabled by inter-zone failures. The *Create Path* packet carries in its body a route that consists of a list of border nodes along the path between the source and destination nodes of the VP. The *Create Path* packet is generated by one of the two border nodes that share the failed link. The *Create Path* packet is terminated by the border node of the last zone that the old and new paths have in common.

Now referring back to Figure 3, the two end-points of the failed inter-zone link, which are border nodes "Node 1.1" 240 and "Node 2.2" 310, detect the failure on "Link 0" 300 and send one or more *Link Down* indications to all end-nodes affected by the failure. An end-node is any node that terminates a VP within that zone, including proxy source and destination nodes. In this example, the two end nodes are "Node 1.6" 200 and "Node 2.3" 320.

"Node 2.2" 310, a master border node realizes that the failed link has a pre-planned alternate path, so it formats the following *Create Path* request of Table 4 and sends it to "Node 2.6" 315:

Field	Contents
Origin	2.2
Target	2.6
VPID	0x20060001
PathIndex	0
Path	2.6, 1.7

Table 4

When the *Create Path* request arrives at node "Node 2.6" 315, it simply increments the *PathIndex* field and forwards the modified request to "Node 1.7" 230, the next node along the path. The *Create Path* request also initiates path establishment within its zone by sending a *Restore Path* request to node 2.3, the Destination node of the VP.

Failure Restoration

Once a node has detected a failure on one of its links, either through a local loss of signal defect or a received alarm indication signal, the node scans its VP table looking for entries that have the failed link in their path. When the node finds such an entry, the node releases all link bandwidth used by the VP. Then, if the node happened to be the source node of the VP, or a proxy border node, the node changes its state to "restoring" and places the nodes VP on a list of VPs to be restored. Otherwise if the node is not the source node or a proxy border node the state of the VP is changed to "down," and a timer is started to delete the node from the database. If a corresponding *Restore Path* request is not received from the origin node within a certain timeout period the timer is started.

The VP list that was created in the previous step is rank-ordered by QoS, ensuring that VPs with a higher QoS are restored first. Each entry in the list contains, among other things, the ID of the VP, Source and Destination nodes of the VP, configured QoS level, and required bandwidth.

When the *Create Path* request arrives at node 230 (Node 1.7) 230, the last node in the specified path, Node 1.7 sends a *Change Target* request to node 200 (Node 1.6), node 200 (Node 1.6) being the source node of the VP. "Node 1.7" 230 does not forward the *Create Path* request since there are no other entries in the path. Upon receiving the *Change Target*

request from node 230 (Node 1.7), node 200 (Node 1.6) formats and sends a *Restore Path* request to node 200 (Node 1.6).

Once an acceptable instance of the *Restore Path* request has reached node 230 (Node 1.7) 230, node 230 (Node 1.7) sends a *Create Path* response to node 315 (Node 2.6). The

5   response, as illustrated in Table 5, contains a list of ports allocated for the VP on the inter-zone link. In this example, node 230 (Node 1.7) allocates port 4 and port 6.

Field	Contents
Origin	1.7
Target	2.6
VPID	0x20060001
PathIndex	0
Path	2.6, 1.7
Ports	4, 6

Table 5

10

When the positive response reaches node 315 (Node 2.6), the sub-path in "Zone 2" 110 connects to the ports specified in the response. Node 315 (Node 2.6) then forwards the response to Node 310 (Node 2.2). Node 310 (Node 2.2) is the master border node that generated the *Create Path* request.

15

Two Hop Inter-zone Alternate Path

In this example, the preplanned alternate path passes through a transit zone. The transit zone 120 is also referred to as Zone 3. A transit zone is defined as a zone that contains one or more tandem nodes used by a particular VP, with the transit node neither originating

20   nor terminating that VP. The first two steps are the same as described in the previous example, except for the path shown in the *Create Path* message. The alternate path in this example is:

Node 335 (Node 2.1)→ Node 365 (Node 3.5)→ Node 345 (Node 3.1)→ Node 260 (Node 1.3)

25

When the *Create Path* message arrives at a transit zone 120 (Zone 3), the entry border node 365 (Node 3.5) forwards the request to the exit border node 345 (Node 3.1). When the *Create Path* message reaches node 345 (Node 3.1), path establishment is initiated within the zone by sending a *Restore Path* request to node 365 (Node 3.5). Node 345 (Node 3.1)

5   forwards the *Create Path* request to node 260 (Node 1.3), the next node along the specified path. When the message finally reaches node 260 (Node 1.3) in the target zone, node 200 (Node 1.6) receives a *Change Target* request. Node 200 (Node 1.6) being the source node of the VP. If zone 100 (Zone 1) were a transit zone, the *Change Target* request would be sent to a proxy source node instead.

10   The *Change Target* request triggers node 200 (Node 1.6) to send a *Restore Path* request to node 260 (Node 1.3). When an acceptable instance of the *Restore Path* request arrives at node 260 (Node 1.3), node 260 (Node 1.3) formats and sends a *Create Path* response to node 345 (Node 3.1). The response, illustrated in Table 6, contains a list of ports allocated for the path on link 370 (Link 2). In this particular example, link 370 (Link 2)

15   includes a port 3 and a port 6.

Field	Contents
Origin	1.3
Target	3.1
VPID	0x20060001
PathIndex	2
Path	2.1, 3.5, 3.1, 1.3
Ports	3, 6

Table 6

20   When the *Create Path* response reaches node 345 (Node 3.1), the create path response allocates the specified ports on link 370 (Link 2) and forwards a modified version of the response to node 3.5 (Node 3.5), as illustrated in Table 7.

Field	Contents
Origin	3.1
Target	3.5
VPID	0x20060001
PathIndex	1
Path	2.1, 3.5, 3.1, 1.3
Ports	Not used on intra-zone links

Table 7

Node 365 (Node 3.5), upon receiving the above response, allocates the required number of ports on "Link 3" 340, appends the required number of ports to the response, and sends the response to node 335 (Node 2.1), as illustrated in Table 8.

Field	Contents
Origin	3.5
Target	2.1
VPID	0x20060001
PathIndex	0
Path	2.1, 3.5, 3.1, 1.3
Ports	7, 9

Table 8

Node 335 (Node 2.1), upon receiving the response from node 365 (Node 3.5), allocates the specified port 7 and port 9 on link 340 (Link 3) and connects them to the sub-path in "Zone 2" 110. "Node 2.1" 335 also forwards the response to "Node 2.2" 310 which is the master border node, and thus completing the loop.

An Example Computing and Network Environment

Figure 6 is a block diagram illustrating a network environment in which a system according to the present invention may be practiced. As is illustrated in Figure 6, network

600, such as a private wide area network (WAN) or the Internet, includes a number of networked servers 610(1)-(N) that are accessible by client computers 620(1)-(N). Communication between client computers 620(1)-(N) and servers 610(1)-(N) typically occurs over a publicly accessible network, such as a public switched telephone network (PSTN), a

5   DSL connection, a cable modem connection or large bandwidth trunks (e.g., communications channels providing T1 or OC3 service). Client computers 620(1)-(N) access servers 610(1)-(N) through, for example, a service provider. This might be, for example, an Internet Service Provider (ISP) such as America On-Line™, Prodigy™, CompuServe™ or the like. Access is typically had by executing application specific software (e.g., network connection software

10   and a browser) on the given one of client computers 620(1)-(N).

One or more of client computers 620(1)-(N) and/or one or more of servers 610(1)-(N) may be, for example, a computer system of any appropriate design, in general, including a mainframe, a mini-computer or a personal computer system. Such a computer system typically includes a system unit having a system processor and associated volatile and non-

15   volatile memory, one or more display monitors and keyboards, one or more diskette drives, one or more fixed disk storage devices and one or more printers. These computer systems are typically information handling systems which are designed to provide computing power to one or more users, either locally or remotely. Such a computer system may also include one or a plurality of I/O devices (i.e., peripheral devices) which are coupled to the system

20   processor and which perform specialized functions. Examples of I/O devices include modems, sound and video devices and specialized communication devices. Mass storage devices such as hard disks, CD-ROM drives and magneto-optical drives may also be provided, either as an integrated or peripheral device. One such example computer system, discussed in terms of client computers 620(1)-(N) is shown in detail in Figure 6.

25   Figure 7 depicts a block diagram of a computer system 710 suitable for implementing the present invention, and example of one or more of client computers 620(1)-(N). Computer system 710 includes a bus 712 which interconnects major subsystems of computer system 710 such as a central processor 714, a system memory 716 (typically RAM, but which may also include ROM, flash RAM, or the like), an input/output controller 718, an external audio

30   device such as a speaker system 720 via an audio output interface 722, an external device such as a display screen 724 via display adapter 726, serial ports 728 and 730, a keyboard 732 (interfaced with a keyboard controller 733), a storage interface 734, a floppy disk drive

736 operative to receive a floppy disk 738, and a CD-ROM drive 740 operative to receive a CD-ROM 742. Also included are a mouse 746 (or other point-and-click device, coupled to bus 712 via serial port 728), a modem 747 (coupled to bus 712 via serial port 730) and a network interface 748 (coupled directly to bus 712).

5   Bus 712 allows data communication between central processor 714 and system memory 716, which may include both read only memory (ROM) or flash memory (neither shown), and random access memory (RAM) (not shown), as previously noted. The RAM is generally the main memory into which the operating system and application programs are loaded and typically affords at least 66 megabytes of memory space. The ROM or flash

10   memory may contain, among other code, the Basic Input-Output system (BIOS) which controls basic hardware operation such as the interaction with peripheral components. Applications resident with computer system 710 are generally stored on and accessed via a computer readable medium, such as a hard disk drive (e.g., fixed disk 744), an optical drive (e.g., CD-ROM drive 740), floppy disk unit 736 or other storage medium. Additionally,

15   applications may be in the form of electronic signals modulated in accordance with the application and data communication technology when accessed via network modem 747 or interface 748.

Storage interface 734, as with the other storage interfaces of computer system 710, may connect to a standard computer readable medium for storage and/or retrieval of

20   information, such as a fixed disk drive 744. Fixed disk drive 744 may be a part of computer system 710 or may be separate and accessed through other interface systems. Many other devices can be connected such as a mouse 746 connected to bus 712 via serial port 728, a modem 747 connected to bus 712 via serial port 730 and a network interface 748 connected directly to bus 712. Modem 747 may provide a direct connection to a remote server via a

25   telephone link or to the Internet via an internet service provider (ISP). Network interface 748 may provide a direct connection to a remote server via a direct network link to the Internet via a POP (point of presence). Network interface 748 may provide such connection using wireless techniques, including digital cellular telephone connection, Cellular Digital Packet Data (CDPD) connection, digital satellite data connection or the like.

30   Many other devices or subsystems (not shown) may be connected in a similar manner (e.g., bar code readers, document scanners, digital cameras and so on). Conversely, it is not necessary for all of the devices shown in Figure 7 to be present to practice the present

invention. The devices and subsystems may be interconnected in different ways from that shown in Figure 7. The operation of a computer system such as that shown in Figure 7 is readily known in the art and is not discussed in detail in this application. Code to implement the present invention may be stored in computer-readable storage media such as one or more

5 of system memory 716, fixed disk 744, CD-ROM 742, or floppy disk 738. Additionally, computer system 710 may be any kind of computing device, and so includes personal data assistants (PDAs), network appliance, X-window terminal or other such computing device. The operating system provided on computer system 710 may be MS-DOS®, MS-WINDOWS®, OS/2®, UNIX®, Linux® or other known operating system. Computer system

10 710 also supports a number of Internet access tools, including, for example, an HTTP-compliant web browser having a JavaScript interpreter, such as Netscape Navigator® 8.0, Microsoft Explorer® 8.0 and the like.

Moreover, regarding the signals described herein, those skilled in the art will recognize that a signal may be directly transmitted from a first block to a second block, or a

15 signal may be modified (e.g., amplified, attenuated, delayed, latched, buffered, inverted, filtered or otherwise modified) between the blocks. Although the signals of the above described embodiment are characterized as transmitted from one block to the next, other embodiments of the present invention may include modified signals in place of such directly transmitted signals as long as the informational and/or functional aspect of the signal is

20 transmitted between blocks. To some extent, a signal input at a second block may be conceptualized as a second signal derived from a first signal output from a first block due to physical limitations of the circuitry involved (e.g., there will inevitably be some attenuation and delay). Therefore, as used herein, a second signal derived from a first signal includes the first signal or any modifications to the first signal, whether due to circuit limitations or due to

25 passage through other circuit elements which do not change the informational and/or final functional aspect of the first signal.

The foregoing described embodiment wherein the different components are contained within different other components (e.g., the various elements shown as components of computer system 710). It is to be understood that such depicted architectures are merely

30 examples, and that in fact many other architectures can be implemented which achieve the same functionality. In an abstract, but still definite sense, any arrangement of components to achieve the same functionality is effectively "associated" such that the desired functionality is

achieved. Hence, any two components herein combined to achieve a particular functionality can be seen as "associated with" each other such that the desired functionality is achieved, irrespective of architectures or intermediate components. Likewise, any two components so associated can also be viewed as being "operably connected", or "operably coupled", to each

5   other to achieve the desired functionality.

Figure 8 is a block diagram depicting a network 800 in which computer system 810 is coupled to an internetwork 810, which is coupled, in turn, to client systems 820 and 830, as well as a server 840. Internetwork 810 (e.g., the Internet) is also capable of coupling client systems 820 and 830, and server 840 to one another. With reference to computer system 810,

10   modem 847, network interface 848 or some other method can be used to provide connectivity from computer system 810 to internetwork 810. Computer system 810, client system 820 and client system 830 are able to access information on server 840 using, for example, a web browser (not shown). Such a web browser allows computer system 810, as well as client systems 820 and 830, to access data on server 840 representing the pages of a website hosted

15   on server 840. Protocols for exchanging data via the Internet are well known to those skilled in the art. Although Figure 8 depicts the use of the Internet for exchanging data, the present invention is not limited to the Internet or any particular network-based environment.

Referring to Figures 6, 7 and 8, a browser running on computer system 810 employs a TCP/IP connection to pass a request to server 840, which can run an HTTP "service" (e.g.,

20   under the WINDOWS® operating system) or a "daemon" (e.g., under the UNIX® operating system), for example. Such a request can be processed , for example, by contacting an HTTP server employing a protocol that can be used to communicate between the HTTP server and the client computer. The HTTP server then responds to the protocol, typically by sending a "web page" formatted as an HTML file. The browser interprets the HTML file and may form

25   a visual representation of the same using local resources (e.g., fonts and colors).

Although the present invention has been described in connection with several embodiments, the invention is not intended to be limited to the specific forms set forth herein, but on the contrary, it is intended to cover such alternatives, modifications, and equivalents as can be reasonably included within the scope of the invention as defined by the appended

30   claims.